| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 1 | BRS | L1 | 9833 | rule near5 (type or kind or define or defininging or definition or content or categor$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:13 |
| 2 | BRS | L2 | 34318 | regulat$5 near5 (type or kind or define or defininging or definition or content or categor$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:11 |
| 3 | BRS | L3 | 5985 | criteri$3 near5 (type or kind or define or defininging or definition or content or categor$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:12 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 4 | BRS | L4 | 124393 | requir$5 near5 (type or kind or define) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:17 |
| 5 | BRS | L5 | 4997 | requir$5 near5 (definining or definition) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:17 |
| 6 | BRS | L6 | 27961 | requir$5 near5 (content or categor$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:18 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 7 | BRS | L7 | 1022 | rule near5 attribut$5 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:19 |
| 8 | BRS | L8 | 287 | regulat$5 near5 attribut$5 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:19 |
| 9 | BRS | L9 | 464 | criteri$3 near5 attribut$5 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:19 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 10 | BRS | L10 | 3366 | requir$5 near5 attribut$5 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:20 |
| 11 | BRS | L11 | 74783 | data near5 (allow or allowed or allowing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:27 |
| 12 | BRS | L12 | 74783 | datum near5 (allow or allowed or allowing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:34 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 13 | BRS | L13 | 5463 | (data or datum or database) near5 (authorize or authorized or authorization or authorizing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:35 |
| 14 | BRS | L14 | 2757 | (data or datum or database) near5 (auhtenticate or authenticated or authentication or authenticating) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:35 |
| 15 | BRS | L15 | 15643 | (data or datum or database) near5 (verify or verifying or verified or verification) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:38 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 16 | BRS | L16 | 35493 | (data or datum or database) near5 (permit or permitting or permitted or permission) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:44 |
| 17 | BRS | L17 | 3745 | database near5 (allow or allowed or allowing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:44 |
| 18 | BRS | L18 | 78704 | data near5 (enable or enabled or enabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:49 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 19 | BRS | L19 | 1915 | database near5 (enable or enabled or enabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:49 |
| 20 | BRS | L20 | 78704 | datum near5 (enable or enabled or enabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:54 |
| 21 | BRS | L21 | 9152 | (data or datum or database) near5 (disable or disabled or disabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 15:57 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 22 | BRS | L22 | 15427 | (data or datum or database) near5 inhibit$4 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:06 |
| 23 | BRS | L23 | 45408 | data near5 (prevent or prevented or preventing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:25 |
| 24 | BRS | L24 | 588 | database near5 (prevent or prevented or preventing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:26 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 25 | BRS | L25 | 45408 | datum near5 (prevent or prevented or preventing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:31 |
| 26 | BRS | L26 | 389 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (11 or 12 or 13 or 14 or 15 or 16 or 17 or 18 or 19 or 20) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:18 |
| 27 | BRS | L27 | 45 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (21 or 22 or 23 or 24 or 25) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:47 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 28 | BRS | L28 | 49705 | (information or record or file) near5 (allow or allowed or allowing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:39 |
| 29 | BRS | L29 | 47036 | (information or record or file) near5 (enable or enabled or enabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:42 |
| 30 | BRS | L30 | 4739 | (information or record or file) near5 (authorize or authorized or authorization or authorizing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:43 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 31 | BRS | L31 | 2968 | (information or record or file) near5 (auhtenticate or authenticated or authentication or authenticating) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:43 |
| 32 | BRS | L32 | 9911 | (information or record or file) near5 (verify or verifying or verified or verification) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:44 |
| 33 | BRS | L33 | 24536 | (information or record or file) near5 (permit or permitting or permitted or permission) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:47 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 34 | BRS | L34 | 3131 | (information or record or file) near5 (disable or disabled or disabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:48 |
| 35 | BRS | L35 | 32414 | (information or record or file) near5 (inhibit$4 or prevent$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:53 |
| 36 | BRS | L36 | 258 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (28 or 29 or 30 or 31 or 32 or 33) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:53 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 37 | BRS | L37 | 21 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (34 or 35) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:55 |
| 38 | BRS | L38 | 67122 | (application or program or software or game) near5 (allow or allowed or allowing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 16:59 |
| 39 | BRS | L39 | 54947 | (application or program or software or game) near5 (enable or enabled or enabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:03 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 40 | BRS | L40 | 2910 | (application or program or software or game) near5 (authorize or authorized or authorization or authorizing) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:03 |
| 41 | BRS | L41 | 1331 | (application or program or software or game) near5 (auhtenticate or authenticated or authentication or authenticating) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:04 |
| 42 | BRS | L42 | 7486 | (application or program or software or game) near5 (verify or verifying or verified or verification) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:05 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 43 | BRS | L43 | 37980 | (application or program or software or game) near5 (permit or permitting or permitted or permission) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:08 |
| 44 | BRS | L44 | 4847 | (application or program or software or game) near5 (disable or disabled or disabling) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:09 |
| 45 | BRS | L45 | 50387 | (application or program or software or game) near5 (inhibit$4 or prevent$4) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:15 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 46 | BRS | L46 | 310 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (38 or 39 or 40 or 41 or 42 or 43) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:15 |
| 47 | BRS | L47 | 39 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near15 (44 or 45) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:16 |
| 48 | BRS | L48 | 15 | (26 or 36 or 46) and (27 or 37 or 47) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:20 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 49 | BRS | L49 | 390 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near5 remot$3 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:19 |
| 50 | BRS | L50 | 423 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near5 central$3 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:19 |
| 51 | BRS | L51 | 440 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near5 center | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:19 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 52 | BRS | L52 | 3556 | (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9 or 10) near5 separat$4 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:20 |
| 53 | BRS | L53 | 66 | (26 or 36 or 46 or 27 or 37 or 47) and (49 or 50 or 51 or 52) | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:23 |
| 54 | BRS | L54 | 81 | 53 or 48<br>Scanned Ti, Ab, Kwic all | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:54 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 55 | BRS | L55 | 3834317 | @pd<19710101 | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:55 |
| 56 | BRS | L56 | 2744 | (705/50 or 705/51 or 705/1 or 707/1 or 707/9 or 707/104).ccls. | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:57 |
| 57 | BRS | L57 | 23 | 55 and 56<br>Scanned Ti all | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:58 |

| | Type | L # | Hits | Search Text | DBs | Time Stamp |
|---|---|---|---|---|---|---|
| 58 | BRS | L58 | 4 | ("5933498" or "6098172").pn.<br><br>From prior search. | USPAT; USOCR; EPO; JPO; Derwent; IBM TDB | 2001/01/05 17:59 |

| | Document ID | Issue Date | Inventor | Current OR | Current XRef | Pages |
|---|---|---|---|---|---|---|
| 1 | JP 03276936 A | 19911209 | KIHARA, YOICHI | | 340/825.06 | 1 |
| 2 | JP 07219900 A | 19950818 | NAKASUJI, MASATAKA , KAWAI, SHIYOUICHI , et al. | | | 11 |
| 3 | US 5875330 A | 19990223 | Goti, Juan Carlos | 717/1 | | 44 |
| 4 | US 5892900 A | 19990406 | Ginter, Karl L. , et al. | 713/200 | 713/201 | 359 |
| 5 | US 6112181 A | 20000829 | Shear, Victor H. , et al. | 705/1 | | 149 |

L54 results

| | Document ID | Issue Date | Inventor | Current OR | Current XRef | Pages |
|---|---|---|---|---|---|---|
| 1 | US 5933498 A | 19990803 | Schneck, Paul B. , et al. | 705/54 | | 51 |
| 2 | US 6098172 A | 20000801 | Coss, Michael John , et al. | 713/201 | | 20 |

L58 results

DIALOG 05 JANUARY 2001

File 2:INSPEC 1969-2000/Dec W3 (c) 2000 Institution of Electrical Engineers
File 6:NTIS 1964-2001/Jan W4 Comp&distr 2000 NTIS, Intl Cpyrght All Right
File 7:Social SciSearch(R) 1972-2001/Jan W1 (c) 2001 Inst for Sci Info
File 8:Ei Compendex(R) 1970-2000/Dec W2 (c) 2000 Engineering Info. Inc.
File 9:Business & Industry(R) Jul/1994-2001/Jan 04 (c) 2001 Resp. DB Svcs.
File 14:Mechanical Engineering Abs 1973-2001/Jan (c) 2001 Cambridge Sci Abs
File 15:ABI/Inform(R) 1971-2001/Jan 05 (c) 2001 Bell & Howell
File 16:Gale Group PROMT(R) 1990-2001/Jan 04 (c) 2001 The Gale Group
File 20:World Reporter 1997-2001/Jan 05 (c) 2001 The Dialog Corporation
File 34:SciSearch(R) Cited Ref Sci 1990-2001/Jan W1 (c) 2001 Inst for Sci Info
File 35:Dissertation Abstracts Online 1861-2000/Dec (c) 2000 UMI
File 42:PHARMACEUTICAL NEWS INDEX 1974-2000/Dec W3 (c) 2000 Bell & Howell
File 43:Health News Daily 1990-2001/Dec 22 (c) 2001 F-D-C reports Inc.
File 63:Transport Res(TRIS) 1970-2001/Dec (c) fmt only 2001 Dialog Corp.
File 65:Inside Conferences 1993-2001/Dec W5 (c) 2001 BLDSC all rts. reserv.
File 74:Int.Pharm.Abs. 1970-2000/Oct (c) 2000 Amer.Soc.of Health-System Pharm.
File 77:Conference Papers Index 1973-2000/Nov (c) 2000 Cambridge Sci Abs
File 80:TGG Aerospace/Def.Mkts(R) 1986-2001/Jan 04 (c) 2001 The Gale Group
File 94:JICST-EPlus 1985-2000/Dec W5 (c)2000 Japan Science and Tech Corp(JST)
File 99:Wilson Appl. Sci & Tech Abs 1983-2000/Nov (c) 2000 The HW Wilson Co.
File 108:AEROSPACE DATABASE 1962-2000/DEC (c) 2000 AIAA
File 129:PHIND(Archival) 1980-2001/Dec W5 (c) 2001 PJB Publications, Ltd.
File 130:PHIND(Daily & Current) 2001/Jan 05 (c) 2001 PJB Publications,Ltd.
File 148:Gale Group Trade & Industry DB 1976-2001/Jan 04 (c)2001 The Gale Group
File 149:TGG Health&Wellness DB(SM) 1976-2000/Dec W3 (c) 2001 The Gale Group
File 151:HealthSTAR 1975-2000/Dec (c) format only 2000 The Dialog Corporation
File 155:MEDLINE(R) 1966-2000/Dec W4 (c) format only 2000 Dialog Corporation
File 160:Gale Group PROMT(R) 1972-1989 (c) 1999 The Gale Group
File 169:Insurance Periodicals 1984-1999/Nov 15 (c) 1999 NILS Publishing Co.
File 233:Internet & Personal Comp. Abs. 1981-2000/Dec (c) 2000 Info. Today Inc.
File 256:SoftBase:Reviews,Companies&Prods. 85-2001/Dec (c)2001 Info.Sources Inc
File 267:Finance & Banking Newsletters 2000/Dec 29 (c) 2000 The Dialog Corp.
File 268:Banking Information Source 1981-2001/Dec W5 (c) 2001 Bell & Howell
File 275:Gale Group Computer DB(TM) 1983-2001/Jan 02 (c) 2001 The Gale Group
File 278:Microcomputer Software Guide 2000/Dec (c) 2000 Reed Elsevier Inc.
File 347:JAPIO Oct 1976-2000/Jul(UPDATED 001114) (c) 2000 JPO & JAPIO
File 348:EUROPEAN PATENTS 1978-2000/DEC W05 (c) 2001 European Patent Office
File 349:PCT Fulltext 1983-2000/UB=20001228, UT=20001214 (c) 2000 WIPO/MicroPat
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec (c) 1998 Inst for Sci Info
File 442:AMA Journals 1982-2000/Oct B3 (c)2000 Amer Med Assn -FARS/DARS apply
File 444:New England Journal of Med. 1985-2001/Dec W1 (c) 2001 Mass. Med. Soc.
File 455:Drug News & Perspectives 1992-2001/Dec (c) 2001 Prous Science
File 473:Financial Times Abstracts 1998-2001/Jan 04 (c) 2001 The New York Times
File 474:New York Times Abs 1969-2001/Jan 04 (c) 2001 The New York Times

File 475:Wall Street Journal Abs 1973-2001/Jan 04 (c) 2001 The New York Times
File 583:Gale Group Globalbase(TM) 1986-2001/Dec 28 (c) 2001 The Gale Group
File 608:KR/T Bus.News. 1992-2001/Jan 05 (c)2001 Knight Ridder/Tribune Bus News
File 621:Gale Group New Prod.Annou.(R) 1985-2001/Jan 04 (c) 2001 The Gale Group
File 623:Business Week 1985-2000/Dec W2 (c) 2000 The McGraw-Hill Companies Inc
File 624:McGraw-Hill Publications 1985-2001/Jan 03 (c) 2001 McGraw-Hill Co. Inc
File 625:American Banker Publications 1981-2001/Jan 05 (c) 2001 American Banker
File 634:San Jose Mercury Jun 1985-2000/Dec 31 (c) 2001 San Jose Mercury News
File 635:Business Dateline(R) 1985-2001/Jan 05 (c) 2001 Bell & Howell
File 636:Gale Group Newsletter DB(TM) 1987-2001/Jan 04 (c) 2001 The Gale Group
File 637:Journal of Commerce 1986-2001/Jan 05 (c) 2001 Journal of Commerce Inc
File 810:Business Wire 1986-1999/Feb 28 (c) 1999 Business Wire
File 813:PR Newswire 1987-1999/Apr 30 (c) 1999 PR Newswire Association Inc

| Set | Items | Description |
|---|---|---|
| S1 | 466285 | (RULE?? OR REGULAT????? OR CRITERI??? OR REQUIR?????) (5N) (TYPE OR KIND OR DEFIN?????? OR CONTENT?? OR CATEGOR????) |
| S2 | 19456 | (RULE?? OR REGULAT????? OR CRITERI??? OR REQUIR?????) (5N) ATTRIBUT????? |
| S3 | 396192 | (DATA OR DATUM OR DATABASE) (5N) (ALLOW???? OR AUTHORIZ??? ??? OR AUTHENTICAT???? OR VERIF????????) |
| S4 | 342817 | (DATA OR DATUM OR DATABASE) (5N) (PERMIT???? OR PERMISSION OR ENABL??? OR DISABL???? OR INHIBIT???? OR PREVENT???) |
| S5 | 1206 | (S1 OR S2) (15N) (S3 OR S4) |
| S6 | 396997 | (INFORMATION OR RECORD?? OR FILE??) (5N) (ALLOW???? OR AUTHORIZ?????? OR AUTHENTICAT???? OR VERIF????????) |
| S7 | 334642 | (INFORMATION OR RECORD?? OR FILE??) (5N) (PERMIT???? OR PERMISSION OR ENABL??? OR DISABL???? OR INHIBIT???? OR PREVENT???) |
| S8 | 1389 | (S1 OR S2) (15N) (S6 OR S7) |
| S9 | 737131 | (APPLICATION OR PROGRAM???? OR SOFTWARE OR GAME?) (5N) (ALLOW???? OR AUTHORIZ?????? OR AUTHENTICAT???? OR VERIF????????) |
| S10 | 702613 | (APPLICATION OR PROGRAM???? OR SOFTWARE OR GAME?) (5N) (PERMIT???? OR PERMISSION OR ENABL??? OR DISABL???? OR INHIBIT???? OR PREVENT???) |
| S11 | 1874 | (S1 OR S2) (15N) (S9 OR S10) |
| S12 | 4609 | (REMOTE?? OR CENTRAL??? OR CENTER OR SEPARATE??) (5N) (S1 OR S2) |
| S13 | 97 | (S5 OR S8 OR S11) AND S12 |
| S14 | 89 | RD S13 (unique items) [Scanned ti,kwic all] |

14/9/5 (Item 5 from file: 15)
DIALOG(R)File 15:ABI/Inform(R) (c) 2001 Bell & Howell. All rts. reserv.

# Relational databases: An accountant's primer

Hooper, Paul; Page, John

01316435 99-65831
Management Accounting v78n4 PP: 48-53 Oct 1996
CODEN: MGACBD ISSN: 0025-1690 JRNL CODE: NAA
DOC TYPE: Journal article LANGUAGE: English LENGTH: 5 Pages
SPECIAL FEATURE: Charts
WORD COUNT: 3313

ABSTRACT: Today, accounting systems data are placed in databases, allowing users to query data without going through the preprogrammed accounting system or learn programming in order to get at the data. There has been a great deal of experimentation with different ways to accomplish the database approach to business processing, and 3 approaches have turned out to be the most common: 1. the hierarchical or tree structure, 2. the network or plex structure, and 3. the relational or table structure. The relational approach is based on tables of data in roles and columns, with operations defined on those tables. Among other things, the relational database allows relationships between tables to be created later, after the data tables have been developed and the data entered. As accounting data processing moves away from centralized mainframe processing it moves toward either decentralized processing, with totally separate databases, or distributed database systems.

TEXT: All information begins as data. The only thing more important than that is how the data are organized when they are stored.

Every day at General Motors 1,183 mainframes process 17 million transactions and borrow $1.7 billion from 700 banks. The IRS receives more than 500 million informational returns each year. VISA has 77.2 million cards generating $60.6 billion dollars in charges, with 25 million cards used in 1,564 automated teller machines in 25 states.

As you can see, acounting data are at the heart of any company's information system, regardless of the level of computer sophistication. Yet, until recently, only trained computer professionals could access computerized data. Users could not access the data directly, so they were not as useful as they could have been.

With the traditional data access approach, queries were difficult. A separate computer program was required for every type of analysis, and it was hard to get access to data for purposes other than those planned for originally and thus preprogrammed into the accounting system. Now accounting systems data are placed in a database. Accounting programs, such as transaction processing and financial reporting, remain much as they were, but the database is accessible directly with tools that the end user can handle. With this approach, virtually anyone can query the database. The user does not have to go through the preprogrammed accounting system or learn programming in order to get to the data.

## BENEFITS AND COSTS OF THE DATABASE APPROACH

Problems with retrieving data in both batch and interactive processing systems using the traditional file approach led to the basic concept behind the database. With the database there is one set of uniquely defined data items, and all computer applications use the same data items that are separate from the applications that use them. This setup allows analysis of the same data across applications. It also means that the applications and the data can be changed independent of each other, so data can be added to, modified, or deleted from the database without the programs using them being affected.

For example, a company may have one set of prices for materials used by inventory control for costing issues, another set of prices in the engineering department used for design of new or revised products, and still another set of prices used by the purchasing department for determining sources. These different sets of prices are updated at different times by different people from different information. Needless to say, the prices probably rarely agree, even though they presumably represent the same thing. The database approach to this problem is to have one set of prices for materials and then have each application use the same information.

The database approach also has simplified applications development. All file systems have the same basic components for file creation, maintenance, transaction processing, and report writing. Once the applications are separated from the data, these programs can be developed just once for all application data. Previously, there had been much duplication of effort in the development of these programs because they were tied to the specific files they used.

But the database approach is not without its costs. The main cost involves coordination. If the same number will uniquely identify a supplier in the ordering system, the accounts payable system, and the inventory system, someone or some group must coordinate the design of these systems. The business cannot allow separate groups to develop systems independently. But the price of coordination can be higher than a company is willing to pay. Also, because each system is not designed unto itself, certain compromises must be accepted in individual components so that the total system will fit together. As a result, each component may not be optimal for a particular task, frequently a concern of users who are more interested in optimizing one specific subsystem, such as inventory, than in optimizing the overall company's operations.

## RELATIONAL DATABASES

Businesses use data items, records, and files to keep track of their operations and accounting data. The most useful way to visualize data items and records is to see them as a table of information called a flat file. The term flat file is used because the information can be viewed in two dimensions: rows (records) and columns (data items) similar to tables of data in a book. Table 1A has a row for each (customer) record in the file and a column for each of the four data items. For convenience, the name of each data item, such as customer number, is at the top of the appropriate column.

For a flat file to be able to store and analyze data, it must have the following characteristics:

1. All items in each column must be the same kind of data, such as a customer number, a customer name, or a customer address.
2. Each column must have its own unique name, separate from all others. In this case, the

names are cust-no, cust-name, cust-addr1, and cust-addr2.

3. All rows must be different in at least one data item from every other row. In other words, two rows of data cannot be exactly alike. If two rows are alike, either they refer to the same customer, so the duplicate can be eliminated, or they refer to two separate customers, in which case there must be one or more data items to distinguish between those customers.

4. Every cell (the intersection of a row and a column) contains only one data item. Thus every customer has exactly one cust-no, one cust-name, and so on.

Having one and only one data item per cell is significant to the design and number of flat files. Suppose a manager is interested in all invoices for a particular customer. Some customers will have only one invoice while others will have two, four, or more. These invoices would not, therefore, fit into a flat file without some modification because each cell must contain only one item. A situation such as multiple invoices for a customer is called a repeating group because there potentially is more than one data item for a customer. Fortunately, repeating groups can be dealt with by forming two flat files. The process of forming additional flat files from the repeating groups is called normalization.

Another name for a flat file is a relation because it represents a relationship among the various data items of the file. In Table 1C, the relation is that all the data items in one record represent one customer. On the other hand, there are two relationships shown in Table 1, one of customers (A) and one of invoices (B). In B, there is one record for each invoice. If there are three invoices for each of six customers, there would be 18 invoices and, thus, 18 records in the invoice file and 18 rows in the table. But there would be only six customer records in the customer relation (flat file).

Some database software can process only flat files, but because it is possible to transform any repeating group into a series of flat files, this restriction is not serious. Databases that are based on flat files and relations are called relational databases. This concept of transforming repeating groups into flat files does create duplication of data. In the invoice example, the customer number is repeated in both the customer file and the invoice file. But relational databases are easy to use and provide flexibility in handling the data, which in most applications outweighs this repetition of data.

## RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

(Table Omitted) Captioned as: Table 1

There has been a great deal of experimentation with different ways to accomplish the database approach to business processing. Three approaches have turned out to be the most common: the hierarchical or tree structure, the network or plex structure, and the relational or table structure.

Hierarchical or tree approach. The first major database was developed at Rockwell International for the purpose of tracking the development of the Apollo space program. The resulting computer program later became known as IMS (Information Management System) when it was sold by IBM. This database had a hierarchical focus: The product (spacecraft) was composed of subassemblies and parts, and each subassembly was composed of further

subassemblies and parts. Eventually, all subassemblies were broken down into their component parts. A complete breakdown of a product into its component parts often is called a bill of materials.

A hierarchical file, such as a bill of materials detailing the components of a manufactured product, has a tree structure relationship between the records of the file. (See Figure 1.) A tree is composed of a hierarchy of elements called nodes. The uppermost level of the hierarchy has only one node, called the root. In our example, this root would correspond to the finished spacecraft. With the exception of the root, every node has another node related to it at a higher level, called its parent. No element can have more than one parent. Each parent can have one or more elements related to it at a lower level, called children; they would be the subassemblies and the parts that compose them. Elements with no nodes in the next level down are called leaves, which would be individual parts with no assembly. Therefore, each node (component) has only one parent (the component it goes into), the root has no parent (because it is complete), and leaves have no children (because they are not assembled). Note that if you look at each element in Level 2 of Figure 1 and think of it as a root, then its children and descendants form a tree. A master file transaction structure also can be thought of as a hierarchical or tree file structure.

The hierarchical approach has the advantage of extremely fast transaction processing. The disadvantages of this approach are that it is extremely complex to set up, often requiring months for the initial project, and it is very difficult to maintain and change as circumstances and data change. Consequently, the hierarchical approach and the IMS program are suitable only for highly structured and extremely highvolume transaction processing environments.

Network or plex approach. The network approach (or plex structure) exists somewhere between the hierarchical and relational approaches in both speed and ease of use and has fallen somewhat out of favor. Users requiring speed choose the hierarchical approach, while those desiring ease of use choose the relational approach. As a result, there is virtually no new development of network DBMSs or new applications using a network DBMS.

If a child in a data relationship has more than one parent, the structure is a network or plex structure. As in tree structures, plex structures may have levels. Figure 2 shows the network structure of a purchasing system with five record types. Each relationship is a parent-child relationship. The purchase order record type is a child of the part (that is, inventory item) record type and a parent of the purchase item record type. A more complex structure is a oneto-many relationship in both directions between part and purchase order. Each part (inventory item) can be purchased using many different purchase orders, and each purchase order can be used for many different parts.

The network or plex structure approach is easier to use (although slower) than the rigid tree or hierarchical approach, and it is the basis for the earlier market success of non-IBM systems such as integrated data management systems (IDMS). The major difficulty with the network approach is that the queries that the system can answer efficiently must be designed into the system. Queries, however, often arise that have not been planned for.

Relational approach. In 1969 at IBM, mathematician E.F. Codd developed the relational theory of data that he proposed as a universal foundation for database systems. Codd's theory formed the basis for all further work in this area. A relational DBMS (RDBMS) satisfies four conditions.

Information-All information in the RDBMS is represented in one way only-as values in tables, which allows users to access, understand, and manipulate data more easily. Each data value should be accessible by the combination of the name of the table in which the data are stored, the name of the column under which stored, and the primary key that identifies the row in which stored.

(Chart Omitted) Captioned as: Figure 1.

(Chart Omitted) Captioned as: Figure 2.

Relational language-The RDBMS requires a data language to define data; define logical displays of data ; manipulate data ; establish rules to prevent errors, such as acceptable values for codes or a required connection between a master file and a transaction file; and maintain authorization of those users able to access the data. This language must be able to process entire tables (not row-by-row) for queries and also for data modification, insertion, and deletion. Independence-Users must not be required to modify queries or application programs if the database has been reorganized and there is no loss of information in the base tables. This would include instances where data are moved (distributed) from one computer to another to be closer to their source or to the location where they are used more often. Integrity management, that is, the maintenance of required links between tables and the validity of values for the data items, should not be duplicated in each application but are implemented by the DBMS

Views-The RDBMS must be able to create logical tables (called views) from the base tables. For example, the information about an individual that a payroll clerk is authorized to look at could be put into a view that the payroll clerk would be able to access. The view could be queried and processed just as a base table would be, even though the original data are not repeated in the view and all of the data are stored only once in the base table.

## ADVANTAGES OF THE RELATIONAL APPROACH

The relational approach is based on tables of data in rows and columns, with operations defined on those tables. Yet these tables must possess the four characteristics of the relational approach described above. The RDBMS (not the user) must ensure that all database tables comply with these requirements. When they do, the RDBMS is able to apply mathematical operations and strict logic to them, which eliminates traditional deficiencies of DBMSs and offers significant practical benefits. The table structure of an RDBMS is simple and familiar. It is general enough to represent most types of data, is independent of any internal computer mechanisms, and it is flexible because the user can restructure tables. Transaction processing is slower than with other approaches, but modifying the structure of files and adding data items (columns) is considerably easier. Also, the relational approach allows relationships between tables to be created later, after the data tables have been developed and the data entered. In the hierarchical and network approaches, allowable queries about the data have to be identified before the database is developed so that the pointers between files and records can be created along with the database.

Data manipulation by an RDBMS is managed by a well-defined, complete set of mathematical operations, which always yield tables as results. With relational operations, data

access no longer needs to be procedural. The user can specify a data request by giving the operations that must be performed on other tables to derive it. The system translates these requests into sets of efficient processing steps. A relational DBMS can accumulate information about the database (such as statistics) in a catalog
to optimize these operations.

## SYSTEMS AND TRENDS

(Photograph Omitted) Captioned as: Relational databases translate requests for information into logical processing steps.

Leading relational programs for mainframes are DB2 for the MVS operating system and SQL/DS for the VM/CMS operating system. An SQL/DS application, for example, can have up to 70 million rows, hundreds of tables, and thousands of columns. SQL/DS is installed on more than 7,500 mainframes and costs more than $100,000 per installation. Other leading relational databases are Oracle and INGRES for minicomputers, dBASE and Paradox for microcomputers.

The mathematical and logical basis of the relational approach makes it a natural candidate for a database standard. A standard based on the relational model would yield the best of all worlds: The products that complied would offer both relational features and compatibility with a defined standard, and the underlying database functions would be the same for all products, regardless of whether they are designed for a single user on a PC or multiple users in more sophisticated systems. In addition, tools such as spreadsheets and word processors do operate on some of these databases. Both the American National Standards Institute and the International Standards Organization have developed standards, and all DBMSs are moving toward them.

RDBMS are moving toward support of distributed databases, which are databases spread throughout the computer systems in a network. One benefit of a distributed database is that local data can be retrieved without any network activity, thus reducing communications costs when compared with a centralized database at a remote site. Another potential advantage is that each database can be sized appropriately for its amount of data, the complexity of user requirements, and the number of users. As the system grows, added demand can be met more easily than with a centralized system by making smaller changes to existing databases or by adding new databases to a network. Current RDBMSs deliver these benefits by allowing a collection of database operations (called a unit of work) to retrieve and update data at a remote site. Future capabilities will add support for a distributed unit of work, which allows a user to access data at multiple locations simultaneously.

RDBMSs are moving toward providing access to the data for applications running on remote computers. This style of distributed computing is called client/server, where the computer providing access to the data is called the database server, and the remote computer requesting the data is called the client.

In client/server, one branch of a company in one location may have primary contact and conduct virtually all transactions with one segment of the company's customers, while other branches work with other customers. It is more efficient from the standpoint of data storage, transaction processing, and data communications if each branch maintains the data files for its

customers while allowing other branches access to the data. This approach is called distributed data processing because the databases are distributed around the operational locations of the firm. If the databases in different branches or divisions are not connected and are kept separate, the company is operating with a decentralized data processing system.

There are four goals for a distributed database system:

A distributed system should appear to each user to be a single, nondistributed system so that queries and transactions that affect distributed data look no different from local queries and transactions.

Each location should have local autonomy and not require the approval of some centralized group for local changes.

A central site should not be required for data storage or processing.

Operation should be continuous.

To achieve these goals, there are several features of a distributed DBMS that should be transparent and of no concern to users. Table 2 summarizes them.

As accounting data processing moves away from centralized mainframe processing it moves toward either decentralized processing, with totally separate databases, or distributed database systems. Proof of the flexibility of the RDBMS is its ability to adjust to the newer, much more complicated ways of dealing with data.

(Table Omitted) Captioned as: Table 2.

Author Affiliation:

Paul Hooper is professor of accounting at the University of Delaware. He can be reached at (302) 831-1795.

John Page is associate professor of accounting at Tulane University. He can be reached at (504) 865-5475.

They are coauthors of Accounting and Information Systems, 4th edition, 1995, published by Prentice Hall, Englewood Cliffs, N.J.

THIS IS THE FULL-TEXT. Copyright National Association of Accountants 1996

14/9/62 (Item 8 from file: 349)
DIALOG(R)File 349:PCT Fulltext (c) 2000 WIPO/MicroPat. All rts. reserv.

00748771 **Image available**
SYSTEM AND METHOD FOR DATA RIGHTS MANAGEMENT
SYSTEME ET PROCEDE DE GESTION DES DROITS EN MATIERE DE DONNEES
Patent Applicant/Assignee: RECIPROCAL INC, 620 Main Street, Buffalo, NY 14202, US, US (Residence), US (Nationality), (For all designated states except: US)
Patent Applicant/Inventor:
LOCKHART Malcolm, 1110 Wellstone Circle, Apex, NC 27502, US
MUSSELWHITE Neal Anthony, 4021 Dutch Harbor Court, Raleigh, NC 27606, US
GRIMES Daniel Gordon, 402 Samara Street, Apex, NC 27502, US
SHARMA Ranjiv, 108 Buck Taylor Trail, Chapel Hill, NC 27516, US
Legal Representative: TOERING Rick A, Cooley Godward LLP, One Freedom Square - Reston Town Center, Suite 1700, 11951 Freedom Drive, Reston, VA 20190-5601, US
Patent and Priority Information (Country, Number, Date):
Patent: WO 200062189 A2 20001019 (WO 0062189)
Application: WO 2000US9654 20000412 (PCT/WO US0009654)
Priority Application: US 99128762 19990412; US 99129139 19990413
Designated States: AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK DM EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG US UZ VN YU ZA ZW
(EP) AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
(OA) BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG
(AP) GH GM KE LS MW SD SL SZ TZ UG ZW
(EA) AM AZ BY KG KZ MD RU TJ TM
Main International Patent Class: G06F-017/00
Publication Language: English
Filing Language: English
Fulltext Word Count: 24374

English Abstract

A system and method for data rights management across multiple data rights management architectures is disclosed. The system and method solves the problems posed by multiple incompatible data rights management architectures. In particular, a data rights management clearing house is provided that generates permits, permit classes, and enables content packaging across multiple data rights management architectures. Consumers may acquire rights to content packaged with different data rights management architecture from the single data rights management clearing house. Additionally, the system and method enables content packagers to package content with multiple data rights management architectures. Finally, the data rights management clearing house provides consumers with a single location from which to manage data access rights and restore data access rights that have been lost.

CLIPPEDIMAGE= JP403276936A
PUB-NO: JP403276936A
DOCUMENT-IDENTIFIER: JP 03276936 A
TITLE: COMMUNICATION SYSTEM
PUBN-DATE: December 9, 1991
INVENTOR-INFORMATION:
NAME
KIHARA, YOICHI
INT-CL_(IPC): H04L012/00; H04L009/00 ; H04L009/10 ; H04L009/12

US-CL-CURRENT: 340/825.06

ABSTRACT:
PURPOSE: To attain the discrimination of the execution of a requirement and to recording medium of a terminal equipment and allowing an information center to acquire the attribute data of lots of unspecified users via a communication channel.

CONSTITUTION: Attribute data such as age and occupation of users are accumulated in storage media 17, 34 provided respectively with a terminal equipment 12 and an external storage medium terminal equipment 25. When the requirement of data transmission/reception is given from the devices 12, 25 to an information center 1, the information center 1 acquires user attribute data 24, 35 stored in the terminal equipment via a communication network 11. Thus, even when lots of unspecified users use a terminal equipment and requirement data transmission/reception of data to the information center, the requirement execution and requirement classification using the user attribute data are attained.

COPYRIGHT: (C)1991,JPO&Japio

14/9/5 (Item 5 from file: 15)
DIALOG(R)File 15:ABI/Inform(R) (c) 2001 Bell & Howell. All rts. reserv.

01316435 99-65831
# Relational databases: An accountant's primer
Hooper, Paul; Page, John
Management Accounting v78n4 PP: 48-53 Oct 1996
CODEN: MGACBD ISSN: 0025-1690 JRNL CODE: NAA
DOC TYPE: Journal article LANGUAGE: English LENGTH: 5 Pages
SPECIAL FEATURE: Charts
WORD COUNT: 3313

ABSTRACT: Today, accounting systems data are placed in databases, allowing users to query data without going through the preprogrammed accounting system or learn programming in order to get at the data. There has been a great deal of experimentation with different ways to accomplish the database approach to business processing, and 3 approaches have turned out to be the most common: 1. the hierarchical or tree structure, 2. the network or plex structure, and 3. the relational or table structure. The relational approach is based on tables of data in roles and columns, with operations defined on those tables. Among other things, the relational database allows relationships between tables to be created later, after the data tables have been developed and the data entered. As accounting data processing moves away from centralized mainframe processing it moves toward either decentralized processing, with totally separate databases, or distributed database systems.

TEXT: All information begins as data. The only thing more important than that is how the data are organized when they are stored.

Every day at General Motors 1,183 mainframes process 17 million transactions and borrow $1.7 billion from 700 banks. The IRS receives more than 500 million informational returns each year. VISA has 77.2 million cards generating $60.6 billion dollars in charges, with 25 million cards used in 1,564 automated teller machines in 25 states.

As you can see, acounting data are at the heart of any company's information system, regardless of the level of computer sophistication. Yet, until recently, only trained computer professionals could access computerized data. Users could not access the data directly, so they were not as useful as they could have been.

With the traditional data access approach, queries were difficult. A separate computer program was required for every type of analysis, and it was hard to get access to data for purposes other than those planned for originally and thus preprogrammed into the accounting system. Now accounting systems data are placed in a database. Accounting programs, such as transaction processing and financial reporting, remain much as they were, but the database is accessible directly with tools that the end user can handle. With this approach, virtually anyone can query the database. The user does not have to go through the preprogrammed accounting system or learn programming in order to get to the data.

## BENEFITS AND COSTS OF THE DATABASE APPROACH

Problems with retrieving data in both batch and interactive processing systems using the traditional file approach led to the basic concept behind the database. With the database there is one set of uniquely defined data items, and all computer applications use the same data items that are separate from the applications that use them. This setup allows analysis of the same data across applications. It also means that the applications and the data can be changed independent of each other, so data can be added to, modified, or deleted from the database without the programs using them being affected.

For example, a company may have one set of prices for materials used by inventory control for costing issues, another set of prices in the engineering department used for design of new or revised products, and still another set of prices used by the purchasing department for determining sources. These different sets of prices are updated at different times by different people from different information. Needless to say, the prices probably rarely agree, even though they presumably represent the same thing. The database approach to this problem is to have one set of prices for materials and then have each application use the same information.

The database approach also has simplified applications development. All file systems have the same basic components for file creation, maintenance, transaction processing, and report writing. Once the applications are separated from the data, these programs can be developed just once for all application data. Previously, there had been much duplication of effort in the development of these programs because they were tied to the specific files they used.

But the database approach is not without its costs. The main cost involves coordination. If the same number will uniquely identify a supplier in the ordering system, the accounts payable system, and the inventory system, someone or some group must coordinate the design of these systems. The business cannot allow separate groups to develop systems independently. But the price of coordination can be higher than a company is willing to pay. Also, because each system is not designed unto itself, certain compromises must be accepted in individual components so that the total system will fit together. As a result, each component may not be optimal for a particular task, frequently a concern of users who are more interested in optimizing one specific subsystem, such as inventory, than in optimizing the overall company's operations.

## RELATIONAL DATABASES

Businesses use data items, records, and files to keep track of their operations and accounting data. The most useful way to visualize data items and records is to see them as a table of information called a flat file. The term flat file is used because the information can be viewed in two dimensions: rows (records) and columns (data items) similar to tables of data in a book. Table 1A has a row for each (customer) record in the file and a column for each of the four data items. For convenience, the name of each data item, such as customer number, is at the top of the appropriate column.

For a flat file to be able to store and analyze data, it must have the following characteristics:

1. All items in each column must be the same kind of data, such as a customer number, a customer name, or a customer address.
2. Each column must have its own unique name, separate from all others. In this case, the

names are cust-no, cust-name, cust-addr1, and cust-addr2.

3. All rows must be different in at least one data item from every other row. In other words, two rows of data cannot be exactly alike. If two rows are alike, either they refer to the same customer, so the duplicate can be eliminated, or they refer to two separate customers, in which case there must be one or more data items to distinguish between those customers.

4. Every cell (the intersection of a row and a column) contains only one data item. Thus every customer has exactly one cust-no, one cust-name, and so on.

Having one and only one data item per cell is significant to the design and number of flat files. Suppose a manager is interested in all invoices for a particular customer. Some customers will have only one invoice while others will have two, four, or more. These invoices would not, therefore, fit into a flat file without some modification because each cell must contain only one item. A situation such as multiple invoices for a customer is called a repeating group because there potentially is more than one data item for a customer. Fortunately, repeating groups can be dealt with by forming two flat files. The process of forming additional flat files from the repeating groups is called normalization.

Another name for a flat file is a relation because it represents a relationship among the various data items of the file. In Table 1C, the relation is that all the data items in one record represent one customer. On the other hand, there are two relationships shown in Table 1, one of customers (A) and one of invoices (B). In B, there is one record for each invoice. If there are three invoices for each of six customers, there would be 18 invoices and, thus, 18 records in the invoice file and 18 rows in the table. But there would be only six customer records in the customer relation (flat file).

Some database software can process only flat files, but because it is possible to transform any repeating group into a series of flat files, this restriction is not serious. Databases that are based on flat files and relations are called relational databases. This concept of transforming repeating groups into flat files does create duplication of data. In the invoice example, the customer number is repeated in both the customer file and the invoice file. But relational databases are easy to use and provide flexibility in handling the data, which in most applications outweighs this repetition of data.

## RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

(Table Omitted) Captioned as: Table 1

There has been a great deal of experimentation with different ways to accomplish the database approach to business processing. Three approaches have turned out to be the most common: the hierarchical or tree structure, the network or plex structure, and the relational or table structure.

Hierarchical or tree approach. The first major database was developed at Rockwell International for the purpose of tracking the development of the Apollo space program. The resulting computer program later became known as IMS (Information Management System) when it was sold by IBM. This database had a hierarchical focus: The product (spacecraft) was composed of subassemblies and parts, and each subassembly was composed of further

subassemblies and parts. Eventually, all subassemblies were broken down into their component parts. A complete breakdown of a product into its component parts often is called a bill of materials.

A hierarchical file, such as a bill of materials detailing the components of a manufactured product, has a tree structure relationship between the records of the file. (See Figure 1.) A tree is composed of a hierarchy of elements called nodes. The uppermost level of the hierarchy has only one node, called the root. In our example, this root would correspond to the finished spacecraft. With the exception of the root, every node has another node related to it at a higher level, called its parent. No element can have more than one parent. Each parent can have one or more elements related to it at a lower level, called children; they would be the subassemblies and the parts that compose them. Elements with no nodes in the next level down are called leaves, which would be individual parts with no assembly. Therefore, each node (component) has only one parent (the component it goes into), the root has no parent (because it is complete), and leaves have no children (because they are not assembled). Note that if you look at each element in Level 2 of Figure 1 and think of it as a root, then its children and descendants form a tree. A master file transaction structure also can be thought of as a hierarchical or tree file structure.

The hierarchical approach has the advantage of extremely fast transaction processing. The disadvantages of this approach are that it is extremely complex to set up, often requiring months for the initial project, and it is very difficult to maintain and change as circumstances and data change. Consequently, the hierarchical approach and the IMS program are suitable only for highly structured and extremely highvolume transaction processing environments.

Network or plex approach. The network approach (or plex structure) exists somewhere between the hierarchical and relational approaches in both speed and ease of use and has fallen somewhat out of favor. Users requiring speed choose the hierarchical approach, while those desiring ease of use choose the relational approach. As a result, there is virtually no new development of network DBMSs or new applications using a network DBMS.

If a child in a data relationship has more than one parent, the structure is a network or plex structure. As in tree structures, plex structures may have levels. Figure 2 shows the network structure of a purchasing system with five record types. Each relationship is a parent-child relationship. The purchase order record type is a child of the part (that is, inventory item) record type and a parent of the purchase item record type. A more complex structure is a oneto-many relationship in both directions between part and purchase order. Each part (inventory item) can be purchased using many different purchase orders, and each purchase order can be used for many different parts.

The network or plex structure approach is easier to use (although slower) than the rigid tree or hierarchical approach, and it is the basis for the earlier market success of non-IBM systems such as integrated data management systems (IDMS). The major difficulty with the network approach is that the queries that the system can answer efficiently must be designed into the system. Queries, however, often arise that have not been planned for.

Relational approach. In 1969 at IBM, mathematician E.F. Codd developed the relational theory of data that he proposed as a universal foundation for database systems. Codd's theory formed the basis for all further work in this area. A relational DBMS (RDBMS) satisfies four conditions.

Information-All information in the RDBMS is represented in one way only-as values in tables, which allows users to access, understand, and manipulate data more easily. Each data value should be accessible by the combination of the name of the table in which the data are stored, the name of the column under which stored, and the primary key that identifies the row in which stored.

(Chart Omitted) Captioned as: Figure 1.

(Chart Omitted) Captioned as: Figure 2.

Relational language-The RDBMS requires a data language to define data; define logical displays of data ; manipulate data ; establish rules to prevent errors, such as acceptable values for codes or a required connection between a master file and a transaction file; and maintain authorization of those users able to access the data. This language must be able to process entire tables (not row-by-row) for queries and also for data modification, insertion, and deletion. Independence-Users must not be required to modify queries or application programs if the database has been reorganized and there is no loss of information in the base tables. This would include instances where data are moved (distributed) from one computer to another to be closer to their source or to the location where they are used more often. Integrity management, that is, the maintenance of required links between tables and the validity of values for the data items, should not be duplicated in each application but are implemented by the DBMS

Views-The RDBMS must be able to create logical tables (called views) from the base tables. For example, the information about an individual that a payroll clerk is authorized to look at could be put into a view that the payroll clerk would be able to access. The view could be queried and processed just as a base table would be, even though the original data are not repeated in the view and all of the data are stored only once in the base table.

## ADVANTAGES OF THE RELATIONAL APPROACH

The relational approach is based on tables of data in rows and columns, with operations defined on those tables. Yet these tables must possess the four characteristics of the relational approach described above. The RDBMS (not the user) must ensure that all database tables comply with these requirements. When they do, the RDBMS is able to apply mathematical operations and strict logic to them, which eliminates traditional deficiencies of DBMSs and offers significant practical benefits. The table structure of an RDBMS is simple and familiar. It is general enough to represent most types of data, is independent of any internal computer mechanisms, and it is flexible because the user can restructure tables. Transaction processing is slower than with other approaches, but modifying the structure of files and adding data items (columns) is considerably easier. Also, the relational approach allows relationships between tables to be created later, after the data tables have been developed and the data entered. In the hierarchical and network approaches, allowable queries about the data have to be identified before the database is developed so that the pointers between files and records can be created along with the database.

Data manipulation by an RDBMS is managed by a well-defined, complete set of mathematical operations, which always yield tables as results. With relational operations, data

access no longer needs to be procedural. The user can specify a data request by giving the operations that must be performed on other tables to derive it. The system translates these requests into sets of efficient processing steps. A relational DBMS can accumulate information about the database (such as statistics) in a catalog to optimize these operations.

## SYSTEMS AND TRENDS

(Photograph Omitted) Captioned as: Relational databases translate requests for information into logical processing steps.

Leading relational programs for mainframes are DB2 for the MVS operating system and SQL/DS for the VM/CMS operating system. An SQL/DS application, for example, can have up to 70 million rows, hundreds of tables, and thousands of columns. SQL/DS is installed on more than 7,500 mainframes and costs more than $100,000 per installation. Other leading relational databases are Oracle and INGRES for minicomputers, dBASE and Paradox for microcomputers.

The mathematical and logical basis of the relational approach makes it a natural candidate for a database standard. A standard based on the relational model would yield the best of all worlds: The products that complied would offer both relational features and compatibility with a defined standard, and the underlying database functions would be the same for all products, regardless of whether they are designed for a single user on a PC or multiple users in more sophisticated systems. In addition, tools such as spreadsheets and word processors do operate on some of these databases. Both the American National Standards Institute and the International Standards Organization have developed standards, and all DBMSs are moving toward them.

RDBMS are moving toward support of distributed databases, which are databases spread throughout the computer systems in a network. One benefit of a distributed database is that local data can be retrieved without any network activity, thus reducing communications costs when compared with a centralized database at a remote site. Another potential advantage is that each database can be sized appropriately for its amount of data, the complexity of user requirements, and the number of users. As the system grows, added demand can be met more easily than with a centralized system by making smaller changes to existing databases or by adding new databases to a network. Current RDBMSs deliver these benefits by allowing a collection of database operations (called a unit of work) to retrieve and update data at a remote site. Future capabilities will add support for a distributed unit of work, which allows a user to access data at multiple locations simultaneously.

RDBMSs are moving toward providing access to the data for applications running on remote computers. This style of distributed computing is called client/server, where the computer providing access to the data is called the database server, and the remote computer requesting the data is called the client.

In client/server, one branch of a company in one location may have primary contact and conduct virtually all transactions with one segment of the company's customers, while other branches work with other customers. It is more efficient from the standpoint of data storage, transaction processing, and data communications if each branch maintains the data files for its

customers while allowing other branches access to the data. This approach is called distributed data processing because the databases are distributed around the operational locations of the firm. If the databases in different branches or divisions are not connected and are kept separate, the company is operating with a decentralized data processing system.

There are four goals for a distributed database system:

A distributed system should appear to each user to be a single, nondistributed system so that queries and transactions that affect distributed data look no different from local queries and transactions.

Each location should have local autonomy and not require the approval of some centralized group for local changes.

A central site should not be required for data storage or processing.

Operation should be continuous.

To achieve these goals, there are several features of a distributed DBMS that should be transparent and of no concern to users. Table 2 summarizes them.

As accounting data processing moves away from centralized mainframe processing it moves toward either decentralized processing, with totally separate databases, or distributed database systems. Proof of the flexibility of the RDBMS is its ability to adjust to the newer, much more complicated ways of dealing with data.

(Table Omitted) Captioned as: Table 2.

Author Affiliation:

Paul Hooper is professor of accounting at the University of Delaware. He can be reached at (302) 831-1795.

John Page is associate professor of accounting at Tulane University. He can be reached at (504) 865-5475.

They are coauthors of Accounting and Information Systems, 4th edition, 1995, published by Prentice Hall, Englewood Cliffs, N.J.

THIS IS THE FULL-TEXT. Copyright National Association of Accountants 1996

CLIPPEDIMAGE= JP407219900A
PUB-NO: JP407219900A
DOCUMENT-IDENTIFIER: JP 07219900 A
TITLE: ELECTRONIC DEVICE
PUBN-DATE: August 18, 1995
INVENTOR-INFORMATION:
NAME
NAKASUJI, MASATAKA
KAWAI, SHIYOUICHI
KIHARA, YOSHIAKI
SAITO, JUNICHI
NISHIOKA, YOKO
INT-CL_(IPC): G06F015/02
ABSTRACT:
PURPOSE: To prevent the contents of secret data from being known by a third required to be secret so as not to permit the contents of data required to be secret to be comprehended.

CONSTITUTION: Date/time data supplied from a clocking part 10 is stored in the date/time data part 111 of RAM 11. Next, a secret flag is read from the data part 115 of RAM 11 by a secret flag storage part 113 and data of a date/ time area and a content area is read by a buffer part 114. Then, the secret flag storage part 113 is referred so that it is judged whether or not the data is the secret one. When it is judged to be secret data, code is converted. At this time, the code being different from the code of original data is set in the buffer part 114. The converted code of the buffer part 114 is transmitted to the display buffer 31 of a CLD driver 3 so that scramble-display being entirely different from original data is executed in a display part 2.

COPYRIGHT: (C)1995,JPO